ليه الTypescript أفضل من الjavaScript عند أصحاب الخبرات، والعكس عند المتعلمين حديثًا، نمسك فى الجزء التاني وهو المتعلمين حديثًا:  
1- الjavaScript بتديك حرية أكتر، مش بتجبرك على نظام معين تشتغل بيه.  
2- الDom فى الjs بالنسبة للمبتدئ هو Super power من الصعب الحصول عليها فى الTypeScript.  
3- قلة المهارة بتخلي المبتدئ يحب الjs؛ لأنها مش بتجبره على نظام معين والكود بالنسباله بيكون أسهل.  
نشوف رأي الخبرات:  
1- الTypescript هي SuperScript بتسمحلك بكتابة الjs بشكلstrongly typed language، والسر كله فى الTypeScript مبني ع الجملة دي، كل النقاط الجاية عنها نابعة عن النقطة دي.  
2- بتسهل عملية الdebugging، بإختصار انت مش محتاج تفتح الكونسول ف المتصفح وتعرف المشكلة جات منين، الTypescript لو انت ماشي بيها صح، هتقولك المشكلة As Aot فى الtext editor.  
3- اي حاجة تشتغل فى الjs تقدر تشغلها بالTypeScript.  
4- الTypeScript ليها منهجية وعقلية قوية، بإختصار الTypeScript جابت كل المصطلحات من الOOP والتوريث والInterface والclass والEs6 الي كلهم بيعلوا الperformance وأجبرتك على فعلها (لو انت ماشي بيها صح)، يعني بدل ما تاخد الjs وتقعد تدور على إتقان مهارات أكتر، هي جات قالتلك اتقني وهتلاقي كل حاجة فيها واكتر.  
-----------  
عيب الTypescript الوحيد او الأشهر هو:  
انها بتاخد وقت اطول فى عملية الCompile، بس متخافش كدة كدة هي سريعة ومش هتحس بفرق كبير.

ما هو الSingle page Application(SPA)؟  
-الإجابة:  
هي تطبيقات ويب تقوم بتحميل صفحة HTML واحدة ويتم تحديث جزء فقط من الصفحة بدلًا من الصفحة بأكملها مع كل نقرة على الماوس، لا تقوم الصفحة بإعادة تحميل أو نقل التحكم إلى صفحة أخرى أثناء العملية، هذا يضمن أداءً عاليًا وتحميل الصفحات بشكل أسرع، تستخدم معظم التطبيقات الحديثة مفهوم SPA؛ حيث ان في SPA يتم إرسال البيانات بالكامل إلى العميل من الخادم في البداية.  
-----------------------------  
مقارنة بسيطة بين الSPA ونظام الMPA (النظام التقليدي):  
فإن الSPA تكون أسرع وأكثر فعالية، حيث لا يحدث تحديث للصفحة في كل مرة نقوم فيها بالضغط على رابط كما يحدث فى الMPA التي لا تعتمد على الAjax او الJson.  
مميزات الSPA:  
1- أسرع.  
2- تغير المحتوى فقط، لا تغير كل الصفحة وكأنها سبورة.  
3- لا تأخذ وقتًا في تحميل المحتوى؛ لأنها تعطي طلب واحد، ولا تخزن كل الطلبات، بل تمحو كل المحتوى الغير مرغوب به حتى يطلبه العميل.  
4- لا تأخذ مساحة كبيرة على المتصفح؛ لأنها لا تخزن كل البيانات دفعة واحدة كما يحدث فى الطريقة التقليدية MPA (Multi Page Application).  
5- تساعد الفريق على التعاون؛ حيث تسمح لمبرمجي الباك اند فى التركيز على الAPI، بينما يقوم مبرمجي الفرونت اند بعمل الUI.  
6- لا تحدث بها الكثير من المشاكل؛ لأنك تعطي طلب واحد للسيرفر وتعود لك بما طلبت فقط، وهذه الطريقة تساعد اصحاب الانترنت البطئ.  
7- تسهل عملية التصحيح (Debugging).  
8- سهولة التصفح والانتقال من صفحة لآخرى.  
--------  
عيوب الSPA:  
1- ضعف الSEO؛ لذا لا يفضل استخدامها مع المواقع التي تتطلب SEO جيد، لكن هذه المشكلة تم حلها فى الكثير من بيئات العمل الخاصة بjavaScript مثل (Angular Universal, Nuxt, Next).  
2- لا تسجل تاريخ التصفح (Browser history).  
3- بها بعض مشاكل الأمان؛ لأن الموقع يكون أسرع يسهل ذلك على المخترق ان يرسل الكثير من السكريبتات التي قد تؤذي الموقع.  
---------  
متى نستخدمها؟:  
تستخدم الSPA لعمل مواقع ديناميكية سريعة سهلة التصفح.  
---------  
أدوات عمل الSPA:  
يمكنك عملSPA بالjavaScript او بالFrameworks المشهورة مثل:  
Angular, React, Vue او غيرها من الFrameworks الآخرى.  
---------  
حل مشكلة الSEO فى الSPA:  
توفر لك الFrameworks المشهورة الكثير من الحلول لهذه المشكلة، وعليك استخدامها حسب الFramework المستخدم:  
Angular > Angular Universal  
React > Next  
Vue > Nuxt  
وهي عبارة عن أطر عمل كما فى الNuxt او الNext او أكواد تضاف كما فى الAngular Universal.

ايه الفرق بين الPure pipe و الImpure pipe؟  
السؤال ده لا بد تتسأله فى اي Angular interview.  
وبرغم سهولة السؤال كتير بيغلطوا فيه.  
هديك إجابة مفهومة ومختصرة بتوضح الفروق الجوهرية وابقى اتعمق مع نفسك:  
1- الImpure pipe بتشتغل لما يحصل تحديث (change detection)، يعني مثلًا انا عندي:  
Array = [1,2,3]  
وعملت function تعمل push رقم 4 بعد ثواني.  
وقولت للpipe بتاعتي احسبي الناتج.  
الpure pipe، بعد مرور الثواني هتطلعلك الناتج ب6.  
الImpure pipe، بعد مرور الثواني هتطلعلك الناتج ب10.  
وده لأن الImpure pipe بتشتغل بطريقة Async وبتعرف الchange detection الي حصل (وده عند اي تغيير، ممكن click او زرار كيبورد مش شرط المثال ده بس).  
على عكس الpure مش بتعمل detection لل change الي حصل.  
2- طيب ايه الفرق فى الكود بين الpure والimpure؟  
فى حالة الimpure هتضيف متغير للdecorator pipe الي عندك اسمه:  
pure وتديه False، بالشكل ده:  
@Pipe({  
name: ‘sumPipe’,  
pure: false  
})  
وفى حالة الpure يكون true او متضيفهوش؛ لأنه by default بيكون pure pipe.  
3- من حيث المشاركة:  
الpure pipe ينفع تشاركها؛ لأنها بتأدي دور ثابت مفيهوش متغيرات، على عكس الimpure مينفعش تشاركها؛ لأنك ممكن تأثر عليها، وبالتالي بتغير من قيمتها.  
4- أمثلة للpure والimpure فى الbulit in pipes:  
1-الpure:  
TitleCasePipe  
DatePipe  
UpperCasePipe  
2-الimpure:  
JsonPipe  
AsyncPipe  
SlicePipe  
وبكدة اكيد عرفت ان الimpure pipes مهمتها الأساسية هي معرفة أخر التغييرات الي تمت، وده يعني انها بتتبع استراتيجية اسمها:  
[#ChangeDetectionStrategy](https://www.linkedin.com/feed/hashtag/?keywords=changedetectionstrategy&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6973278228362743808)  
والاستراتيجية دي تعني: اي تغيير يحدث اتفاعل معاه.

ما هو الAngular؟  
-هو إطار عمل متكامل للويب يعتمد على الtypeScript، يهتم بالجانبين الServer والBrowser  
-هو واحد من أهم وأشهر أطر العمل.  
-يستخدم Angular مئات الالاف من الأشخاص حول العالم.  
-تم انشاء هذا الإطار داخل Google، لصناعة Single page application؛ لبناء المواقع ذات المعمارية المعقدة التي كان يصعب بناءها بالjs والjquery، هذا لا يعني انك لا يمكنك بناء SPA بالjs ولكن سهل الAngular تلك العملية؛ حيث انه يصنع تطبيقات ضخمة قابلة للصيانة؛  
فقد سهل الAngular عملية البحث عن المشاكل وحلها.  
-وهذا يرجع لقوة معمارية الAngular وهو الMVC.  
-مما سهل عملية التعامل والتنظيم داخل إطار العمل، وسهل على المبرمجين عملية البناء؛ فبوضعها داخل معمارية منظمة؛ سهل ذلك بناء التطبيقات الضخمة المعقدة.  
----------------  
هذه مقدمة عن الAngular، وجاري كتابة العديد من المقالات عنها.  
والتي ستبدأ ب(لماذا بنيت أطر العمل الخاصة بالفرونت اند مثل Angular؟)

لماذا تم إنشاء الClient-side frameworks؟  
بصيغة آخرى:  
لماذا تم إنشاء framework للFrontend؟  
نرجع رحلة لقبل 2012، وقبل ظهور العظيم Angular js، كانت مواقع الويب الديناميكية بتتعمل بالjavascript والjQuery.  
طبعا العالم اتطور، واكثر الشركات المعروفة بالApplications المعقدة هي Google؛ فقرروا أنهم يبنوا الAngular js، لتسهيل عملية بناء الApplications، تم إستخدام الAngular js داخل جوجل لفترة قبل عرضها على المبرمجين فى 2012.  
-ملحوظة اول Framework للفرونت تم انشاءه هو Ember js وكان ذلك في عام 2011، لكنه لم يحظى بما حظي به Angularjs من شهرة؛ وذلك لأنه لم يكن قويًا كفاية ولم يحظى بدعم كبير.  
-------------  
ومن هنا بدأت ثورة الFrontend، الFrameworks ولم تكن فقط بديل للjs والjQuery، بل حلت كل المشاكل الخاصة بيهم.  
ومن بعد الAngular js ظهرلنا الReact js، ومن بعدها ظهر الVue js والعديد من الFrameworks الأخرى القوية.  
-------------  
من ضمن الاشياء العظيمة التي قدمها لنا ال Frameworks (المميزات) :  
1- عزل فريق الFrontend عن فريق الBackend:  
بمعنى ان لا يوجد داعي لإنتظار الفرونت ينتهي من التمبلت لإستكمال الBackend، كل تيم يعمل بشكل منفرد، وفى النهاية يحصل تواصل بينهم.  
2- البروجيكتس الكبيرة والمعقدة، كان من الصعب تنفيذها بالjQuery والjs، وكل ما البروجيكت كبر كل ما كان ابطأ والكود غير نظيف، وتظهر الكثير من المشاكل.  
3- الcomponents والmodules، أدى وجودهم لجعل الcoding فى الfrontend ابسط ومنعزل عن الباقي، وبالتبعية تنظيم اكثر لأفراد فريق العمل فى الFrontend؛ على سبيل المثال يقدر اي شخص داخل فريق عمل الفرونت اند العمل على modules معزولة عن الآخرين وبدون الحاجة للتعطيل او حدوث مشاكل.  
4- سرعة حل المشاكل Debugging.  
5- سرعة المواقع.  
6- Routing.  
والعديد من المميزات التي لا تنتهي، ولكن سيتم ذكرها في مقالات أخرى.  
-------------  
وبالتأكيد كل شئ له مزايا، له عيوب ومن بين هذه العيوب كان:  
1- ضعف الSeo، لكن فيما بعد تمت معالجة الأمر بإستخدام SSR Frameworks مبنية على الFrameworks مثل Nuxt, Next والAngular عالجت المشكلة عن طريق الAngular universal.  
2- لا تعمل على كل المتصفحات.  
3- من الممكن ان يحدث تغييرات فى الموقع حسب المتصفح المستخدم.  
4- بعض المواقع الصغيرة لا تستحق إستخدام الFramework، وسيكون الأمر كارثيًا ان اخطأ المبرمج في تقدير حجم الموقع، وسيبذل الكثير من المجهود وهذا مخالف تمامًا للغرض الذي انشأ لأجله الFrameworks.

كيف يعمل الAngular؟  
بعد تثبيت ملف الAngular بيتم انشاء العديد من الFiles والFolders المهمة، في المقال ده هتعرف أهمية أغلب الملفات دي (ان شاء الله).  
------------  
اولًا ملف الpackage json وده مهمته بإختصار يقولك الاسم الخاص بالبروجيكت والأوامر الأساسية + الملفات المثبتة او سواء كdev او مثبتة بشكل عام (dev & build) وده كله تقدر تعدله، والملف ده بيعرف اي حد شغال معاك المكاتب المثبتة إلى جانب الAngular؛ فعند التحميل يكتب أمر npm i في التيرمنال، والتيرمنال يروح يقرأ ملف الpackage ويحمل بناءًا عليه.  
-------------  
ثانيًا Life cycle of angular project:  
هتسألني ايه المصطلح ده؟  
-المصطلح ده بيقولك خريطة تشغيل الموقع، طبعًا الخطوات دي بعد ملف الpackage json، هتقولي ليه محطتهوش الخطوة الأولى فى الlife cycle؟  
-هقولك لأن مهم الPackage json بتقف لحد تثبيت الملفات وبس، لكن الملف بعدين فى الReal project مش هتبقى محتاجه.  
نبدأ بقى فى الlife cycle:  
1- Angular json:  
ده بقى الملف الي بيقول للبروجيكت يبدأ منين؟ ويروح منين؟ ومين يجيب مين؟ والملفات الخارجية المستخدمة وهتلاقي فيه سطرين حلوين، الي هما دول:  
"index": "src/index.html",  
"main": "src/main.ts",  
الاول وده عشان الAngular بتعتمد على الSingle page؛ فالأول بيقول للAngular json هي دي الSingle page بتاعتك.  
2- Main ts:  
وده السطر التاني لو انت مصحصح معايا، وده بيقوله نقطة البداية للبروجيكت، عن طريق Function اسمها: bootstrapModule.  
وبيديله الخطوة الجاية الي معانا: AppModule.  
3- AppModule:  
وده الModule الرئيسي فى البروجيكت، بعيد عن انك ينفع تغيره (مع ان مفيش داعي وده احسن نظام والناس الي معاك هتدعي عليك لو عملت كدة)، بس الAppModule بتاعنا في كام حاجة مهمة كدة:  
-declarations  
-imports  
-providers  
-bootstrap  
الImports دي بتحط فيها الmodules المستخدمة، والproviders بتحط فيها الServices الي عايزها تشتغل ف الmodule، بس دول مش موضوعنا.  
المهم عندنا بالنسبة للcycle الdeclarations والbootstrap ودول ببساطة:  
الdeclarations: بتحط فيها الcomponents التابعين للmodules، ولما بتكون لسه بادئ البروجيكت بيكون فيهاAppComponent وبس.  
الbootstrap: بتحط فيها الcomponent الي هيبدأ يشغلها الmodule، ودي مهمة الbootstrap بتاع الAngular مش التاني بتاع الcss الاتنين مختلفين تمامًا.  
4- AppComponent:  
ودي لما بتدخل عليها بتلاقي فيها دول:  
selector: 'app-root',  
templateUrl: './app.component.html',  
styleUrls: ['./app.component.css']  
الاول ده المهم فى قصتنا، بس هقولك كل واحد فيهم بيعمل ايه؟  
الاول ده اسم الComponent الي اقدر من خلاله اعملها call.  
التاني وده الHtml بتاع الcomponent.  
التالت ايوة شاطر الcss بتاع الcomponent.  
5- index html:  
فاكر صاحبنا الي ف اول رحلتنا فى البروجيكت قولتلك هو الsingle page بتاعنا؟  
اهو الحمدلله بعد الرحلة دي كلها وصلناله، هتلاقيه صفحة Html عادية خالص بس فيها سطر غريب كدة:  
<app-root></app-root>  
معنى كدة ايه؟  
ايوة شاطر، هو عمل call للAppComponent بتاعتنا الي الSelector بتاعها اسمه app-root، وبكدة هيبدأ من الAppComponent، هتروح انت بقى لملف الAppComponent html، وتحذف الكلام الي فيه وهو ده نقطة البداية الي اول ما تنشأ بروجيكت Angular بتكون متحضرالك.  
وكدة يبقى انت عرفت دورة حياة بروجيكت الAngular من الصفر، لحد ما وصل انه بقى بروجيكت قدامك لما بيتنشأ.

ما هو الفرق بين الAngular والAngularjs؟  
من اهم اسئلة الانترفيو.  
معلومات بسيطة مبدئية وبعدين ندخل فى الفروق المهمة:  
✅تاريخ الإنشاء:  
Angular js: 2010  
Angular: 2016  
✅تاريخ الإصدار:  
Angularjs: 2012  
Angular: 2016  
✅تاريخ توقف العمل:  
Angular js: 2022  
Angular: مستمرة وباقية.  
-------------  
✅المعمارية (Architecture):  
-الAngularjs:  
معمارية الMVC وهي الModel-View-Controller:  
الModel: يحتوى على المنطق او المعلومات المطلوبة في البروجيكت (business logic).  
الView: عرض المعلومات الموجودة فى الModel.  
الController: يتحكم بالمعلومات القادمة من الModel.  
-الAngular:  
نفس معمارية الMVC، ولكن بشكل أخر؛ بحيث انه استخدم الComponents والDirectives بدل الController، او يعملوا عمل الController.  
---------------  
✅اللغات المستخدمة:  
Angularjs: javaScript  
Angular: TypeScript  
---------------  
✅دعم الموبايل:  
الAngularjs: لا تدعم الموبايل.  
الAngular: تدعم الموبايل.  
---------------  
✅البنية (Structure):  
الAngularjs: عند بناء التطبيقات الضخمة يصبح الأمر معقدًا.  
الAngular: لن يكون الأمر معقدًا مهما كبر التطبيق وهذا يوضح قوة بنية الAngular وتفوقها على Angularjs.  
---------------  
✅التعبير اللغوي (Expression Syntax):  
الAngularjs: تستخدم العديد من التعبيرات المعقدة التي توجب على المبرمج تذكرها.  
الAngular: لديها ايضًا العديد من التعبيرات المعقدة، لكنها توفر بعضًا منها بإستخدام علامات مثل [] و ()، واللذان يستخدمان فى الdata binding او الTwo way data binding.  
---------------  
✅Design patterns  
الAngularjs: لا تستخدم العديد مما في الAngular مثل: (dependency injection).  
بينما الAngular: تستخدمها.  
---------------  
وهناك المزيد من الفروقات لكن لا يكفيها منشور واحد

الAngular مقابل الReact js ج1  
تنوية:  
هذه مقارنة بين اكبر اتنين technologies في الFrontend web development، وهذه المقارنة للعرض، وليست لتفضيل technology على أخرى.  
✅ الشركة المنشأ:  
Angular: Google.  
React: Facebook.  
✅سنة النشأ:  
Angular: في البداية نشأ كAngular js في 2010 وظهر في عام 2012، لكن بعد ذلك صار Angular يعتمد على الTypeScript وتحول إسمه لAngular وظهر عام 2016.  
React: ظهرت عام 2013.  
✅عبارة عن:  
الAngular: هو Framework لبناء واجهات مواقع ويب متكاملة.  
الReact: هي library تؤدي عمل الFramework ولذلك يعتبرها الكثير من المبرمجين Framework.  
✅لغة البرمجة:  
الAngular: يعتمد على الTypeScript.  
الReact: تعتمد على الjavaScript و الJsx.  
✅دعم الهواتف الذكية:  
الAngular: يدعم الهواتف الذكية.  
الReact: لا تدعم، ولكن لديها مكتبة React Native تستطيع من خلالها بناء تطبيقات للهواتف الذكية.  
✅الBinding:  
الAngular: يستخدم Two way data binding.  
الReact: لها One way data binding.  
✅الDom:  
الAngular: يستخدم الRegular Dom.  
الReact: تستخدم الShadow Dom وهو أسرع بكثير من نظيره.  
✅التكامل:  
الAngular: أكثر تكاملًا؛ حيث أنه Framework يحتوي على كل ما تحتاجه لبناء التطبيق من Routing وReactive programming والعديد من الأدوات المساعدة، ولا يحتاج لمكاتب خارجية كثيرًا لتساعده في البناء.  
الReact: قلما لا تحتاج إلى مكاتب خارجية، من الضروري إستخدام مكاتب خارجية لبناء تطبيق متكامل بها.  
✅الdependency injection:  
الAngular: بالنسبة له فهو نقطة قوة.  
الReact: لا تستخدمها وتعتمد عليك في هذه المهمة، او تستعمل مكتبة طرف ثالث (third-party).  
هناك العديد من النقاط المتبقية للمقارنة، لكن سنوفرها لمنشور آخر.

ما هي الDirectives فى الAngular وما هي أنواعها؟  
سؤال Interview مهم ولازم تتسأله مهما كان الposition الي هتقدم عليه ان شاء الله.  
-التعريف:  
هي Functions تستخدم لمعالجة سلوك(behavior) الDom، ويتم تنفيذها فى الحال ما ان وجدها الCompiler.  
-الأنواع:  
1- الComponents directive:  
هي نوع مميز من الdirectives، طب ليه؟  
لأنها ببساطة بتيجي بالTemplate بتاعها جاهز، المثال عليها هو ال@Component وده Class decorator وDirective: وهو عبارة عن البنية الأساسية لبناء مشروع الAngular، وبيحوي ملف للStyle وملف الTs الي بيكون فيه Functions ومعلومات، وفي ملف تالت للHtml وده هو الخاص بعرض الComponent.  
---------------  
2- الstructural directives:  
وده عبارة عن Directive بيغير الDom كاملًا من حيث الشكل (Layout) والإضافة والحذف، والظهور والتكرار، من الاخر بيستبدل الDom، بناءًا عليه، وهيوضح دلوقتي لما نذكر أمثلة built in عليه وهم:  
ال\*ngIf: وده بيشيل جزء او يخليه ظاهر بناء على الBoolean الي راجعله، لو كان True يظهر لو كان False يختفي.  
ال\*ngFor: بيكرر حسب المطلوب منه لو في داتا راجعاله؛ فهو بينظمها حسب ما تحتاج منه؛ وده بيغير اجزاء كتير فى الDom، لاحظ النقطة دي.  
ال\*ngSwitchCase: برضو بتغير الDom بناء على الحالة الي بترجعله وبيظهر حسب رغبتك Dom او يخفيه.  
بإختصار الstructural directives بتغير الDom وبتتحكم فيه ككل، ولاحظ ان اولها \*.  
----------------  
3- الattribute directives:  
دي من اسمها تقدر تضيف Directive يتحكم فى Attribute معين، وهديك الأمثلة على طول اهو:  
ال[ngClass]: ودي بتتحكم في إضافة وإزالة Class فى الHtml، حسب الboolean الي واصله.  
ال[ngStyle]: ودي بتتحكم في إضافة وإزالة Style فى الHtml، حسب الboolean الي واصله.  
ال[ngModel]: ودي بتتحكم في إستقبال data بone way data binding.  
ال[(ngModel)]: ودي بتتحكم في إستقبال وإرسال data بTwo way data binding.  
----------------  
نضيف سؤالين للمقال:  
1- ما هي الcustom directives؟ (هكمل الإجابة فى مقال منفصل)  
2- ايه الفرق بين اني اضيف Class واعمله display none وبين ال\*ngIf؟  
----------------  
1- الcustom directives دي بتعملها بنفسك عن طريق انشاء directives وينفع تكون structural وينفع تكون attribute، وفي الحالتين بتستخدمهم بين [] لو attr وبتستخدم قبلهم \* لو structural زي الdirectives الbuilt in.  
وان شاء الله فى مقال منفصل يكون شرح وافي لبناء custom directive.  
  
2- الفرق انك لما تضيف Class؛ فهيكون لسه موجود فى الinspect كومنت يظهر الdom المخفي بكل ما في داخله، لكن لما بتعمل \*ngIf وال\*ngIf تحذف الdom وتبدله بكومنت من عند الAngular يوضح سبب الحذف وميظهرش حاجة من الDom.

ما هي الDecorators فى الAngular؟  
(سؤال انترفيو مهم).  
-التعريف:  
الDecorators هي مفهوم أساسي فى الTypescript؛ ولأن الAngular معتمده على الTypeScript؛ فوضعته كحجر أساس فى الAngular.  
هي نوع مميز من الDeclaration؛ غير انها الDesign pattern الأساسي الي بتعتمد عليه الAngular في تعريف خواص عديدة.  
-خصائص:  
1- بتعرفك النوع الي بتتعامل معاه، مثال:  
@ Component  
2- هي Functions بترجع Functions  
3- بتحتوي على Metadata وهو الobject الي بيرجع فيه البيانات الخاصة بالdecorator.  
4- بيشتغل الdecorator مع الruntime.  
5- تساعدك انك تشغل من خلالها Function او تبعت Function او انك تبعت داتا وترجع داتا وإلخ...  
-------------  
ما هي أنواع الDecorators فى الAngular؟  
هما 5 أنواع، فى المنشور ده هنحصر مهمتهم بشكل سريع ونذكر الأنواع؛ لأن شرح كل واحدة فيهم تدخلنا في نقاش وجدال ومقارنة، وده الي هيحصل بإذن الله فى المقالات القادمة.  
  
1- الClass Decorator:  
هو أهم نوع والأكثر إستخدامًا فيهم وهو (highest-level decorator)؛ لأنه بيوضح الغرض من الDecorator وبيوضحلك الClass الي انت شغال فيه اذا كان Component ولا Module، نذكر بعض الأمثلة سريعًا:  
-الPipe، -الComponent، -الNgModule، -الDirective، -الInjectable.  
(ملحوظة قبل اي decorator بنحط علامة @ مهما كان نوعه فى الخمسة).  
-------  
2- الProperty decorator:  
وهو تاني أشهر نوع فى الdecorators وبيساعدك تبعت Property من Component لComponent والعكس او من Directive وهكذا من الأمثلة، وأشهر اتنين فى الفئة دي Input وOutput ودول الي بينقلوا data او Functions من الاب لابن والعكس، وهديك كام مثال على الفئة دي:  
-الViewChild، -والViewChildren، -والContentChild، والContentChildren.  
كلهم مميزين وليهم أدوار قوية في نقل الخواص، لكن مش هيكفيهم حتى منشور لكل واحدة فيهم.  
--------  
3- الaccessor decorator:  
ودول بيستقبلوا parameters زي: name او target.  
واشهر مثالين الGetter والSetter وبيتكتبوا كدة قبل الfunction:  
بيتكتب قبل الget والset الDecorator بالشكل التالي:  
@LogAccessor  
get text(): string {  
return this.\_text;  
}  
  
set text(value: string) {  
this.\_text = value;  
}  
--------  
4- الMethod decorator:  
ودي decorators بتعمل Method، مثال:  
HostListener  
بتعمل بيها Function، وبتستخدم فى الdirective والComponent بنفس الطريقة.  
--------  
5- الParameter decorator:  
في مثال عليها بس قليل لما بيتم استخدامه وهو الInject؛ لأنه اتعوض بطرق آخرى، ومهمته انه يعمل inject للService داخل الconstructor بتا الComponent، بالطريقة دي:  
constructor(@Inject(MyService) myService) {  
console.log(myService); // MyService  
}  
وطبعًا الأفضل استخدام الAccess Modifiers زي الPrivate والPuplic والProtected لما ليهم من خصائص اكبر من مجرد الInject.

ما هي الفروق بين الPromises والObservables فى الAngular؟  
(سؤال متكرر فى انترفيو الAngular).  
-المصدر:  
الPromise جات مع الEs06.  
الObservable: من مكتبة الRxjs.  
----------------------  
المقارنة:  
1- الCall back:  
الpromise ليها حالتين اما Success واما يحصلError.  
الobservable ليها 3 حالات: اما Success او يحصل Error وبعدهم Complete.  
---  
2- الLazy loading:  
الpromise: بتكون not lazy والجملة دي مساوية لجملة (Eager loading).  
الobservable: بتكون lazy.  
---  
3- الEmits:  
الpromiseL: بتعمل Emit لValue واحدة.  
الobservable: بتعمل Emit لأكثر من Value:  
وده بيحصل عن طريق الnext، مثال:  
[observer.next](http://observer.next/)(1)  
[observer.next](http://observer.next/)(2)  
---  
4- الcancellable (الإغلاق):  
الpromise: بتكون not cancellable.  
الobservable: قابلة للإغلاق (cancellable)، وده عن طريق الunsubscribe().  
---  
5- التزامن:  
الpromise: دايمًا ASYNCHRONOUS (غير متزامن).  
الobservable: ممكن تكون ASYNCHRONOUS وممكن SYNCHRONOUS يعني متزامن وغير متزامن.  
---  
6- الrx operators:  
الpromise: لا تقبل الrx operators.  
الobservable: يقبل الrx operators، عن طريق الpipe وبالشكل التالي:  
Obs.pipe(map(value) => value \* 2);  
---  
فى الكومنتس هيكون موجود صور توضح الفروق بينهم.

ترتيب الLife cycle component فى الAngular؟  
من الاسئلة المشهورة فى الانترفيو، ولكن بالتأكيد هيسألك على الاقل عن نقطة من نقاطه، والي غالبًا بيكون عن الأربعة دول:  
AfterViewInit, OnDestroy, Constructor, Oninit، بس ده مش موضوع مقالنا، موضوع مقالنا هو الترتيب، وبإذن الله هنشرح كل hook منهم فى مقال لوحدها، مقالنا هو مقدمة عن الlife cycle، ونبذة بسيطة عن كل hook.  
الترتيب بيكون كالتالي:  
1- الconstructor:  
وده بيشتغل اول حاجة، وبنحتاجه؛ عشان نinject من خلاله الservices ونعطي قيم أساسية بداخله بدون الحاجة لعلامة الnon null assertion(!).  
----  
2- الngOnChanges:  
وده اول حاجة بتشتغل بعد الconstructor، وبيشتغل لما يحصل تغير فى الproperties الخاصة بال@Input.  
----  
3- الngOnInit:  
وده بيشتغل مرة واحدة، وبيهيء الcomponent وبيعمل Set لكل الأساسيات الي بتحتاجها من الcomponent؛ فمثلًا لو عندك data راجعة من Service؛ فبيتعملها Set فى الhook ده وبتساوي من خلالها Object او Array بالdata دي.  
----  
4- الngDoCheck:  
وده بيتشغل لما يحس بأي تغيير فى الcomponent، وتقدر من خلاله تستخدم خوارزمية الchange detection.  
----  
5- الngAfterContentInit:  
وده بيشتغل بعد ما الcontent يبدأ يشتغل فى الview يعني يتعمله render فى الview وليها تعبير تاني اسمه (projected).  
----  
6- الngAfterContentChecked:  
وده بيشتغل فى حالتين:  
-الأولى: لما الcontent يحصل فيه تغييرات.  
-الثانية: لما الngDoCheck يشتغل.  
وممكن تقول ان الحالتين حالة واحدة؛ لأنه لما بيحصل تغيير فى الcontent؛ فالأول بيشتغل الngDoCheck، وبعدها يشتغل الhook الي بنتكلم عنه.  
----  
7- الngAfterViewInit:  
وده بيشتغل لما الView يتعمله built، او لما تعمل built لChild component خاصة بالComponent بتاعتك.  
----  
8- الngAfterViewChecked:  
وده مقارب للhook رقم 6 وبيشتغل لما الview يحصل فيه تغييرات وبعد الngDoCheck.  
----  
9- الngOnDestroy:  
وده بيشتغل لما تسيب الcomponent، وترجع أهميته لأنه بيعمل destroy للداتا لما تروح لcomponent تانية، وبالتالي الابليكشن هيكون أسرع بكتير، وأخف.  
----  
ملحوظة1: (ده الترتيب للlife cycle، لكن مش بيشتغلوا فى سلسلة متتالية Static، بيشتغلوا بطريقة مختلفة، هحط صورة ليها فى الكومنتس).  
ملحوظة2: (وقت الإجابة يفضل تبدل مصطلح البناء او التهيئة بكلمة initialize، عشان إجابتك تكون فورمال اكتر).

ما هو الفرق بين الTemplate driven forms والReactive Forms فى الAngular؟  
الاول لازم تعرف ان في انواع من الForms فى الAngular:  
1- الbasic forms.  
2- الTemplate driven forms.  
3- الReactive Froms.  
4- الDynamic Forms.  
وفى المقال ده مقارنة بين الTemplate driven، والReactive، والأربعة هيتشرحوا بالتفصيل فى مقالات منفصلة ان شاء الله، ندخل فى المقارنة على طول:  
--  
1- الSetup:  
-الTemplate: بتم من اسمها بتم من خلال التمبلت Html عن طريق الdirective.  
-الReactive: بتم من خلال الClass component فى ملف الts.  
-----  
2- الإستخدام:  
-الtemplate: سهل الإستخدام.  
-الReactive: محتاج تدريب.  
-----  
3- الUse case (ضرورة الإستخدام):  
-الtemplate: للسيناريوهات الصغيرة.  
-الReactive: للسيناريوهات المعقدة.  
-----  
4- الData flow:  
-الTemplate: بيكون Asynchronous.  
-الReactive: على العكس بيكون Synchronous.  
-----  
5- الValidation:  
-الTemplate: من خلال الDirectives.  
-الReactive: من خلال الFunctions فى ملف الts.  
-----  
6- شكل الكود:  
-الTemplate: الHtml اكتر وقليل Ts.  
-الReactive: الHtml قليل الTs كتير.  
-----  
7- الtesting:  
-الTemplate: بيكون أصعب.  
-الReactive: بيكون أسهل.  
-----  
8- الData binding:  
-الTemplate: بيكون Two way data binding بيعتمد على ال[(NgModel)]، يعني بيكون (mutable).  
-الReactive: مفيش data binding يعني (immutable).  
----\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ازاي تضيف البوتستراب للAngular؟  
(السؤال ده بيكون انترفيو بنسبة قليلة).  
بس عمومًا تعالى نسأل السؤال المعتاد هل انا محتاج bootstrap؟  
الإجابة:  
نعم؛ لأنك ببساطة بتحتاج Framework css يساعدك فى تسريع عملية بناء الفرونت اند الخاص بيك، وطبعًا أشهر Framework css هو بوتستراب؛ فطبيعي النهاردة اكلمك عنه شوية.  
وسؤال الانترفيو فى الحالة دي عايزك شخص واعي وفاهم لجوانب كتيرة شوية، يلا نبدأ فى الإجابة.  
---------------  
في 3 طرق لإضافة بوتستراب للAngular:  
1- عن طريق الCdn:  
خطوات التركيب:  
-إضافة الCdn لملف الindex html الخاص بالبروجيكت.  
عيوب الطريقة:  
-لا يفضل إستخدام Cdn عمومًا ف اي بروجيكت؛ لأن ممكن يحصل حذف او تعديل او ان الlink نفسه يبقى بطئ، وده هيبدأ الrender للبروجيكت الخاص بيك، وإحتمال يظهر بدون الCss الي داخل الCdn.  
مميزات الطريقة:  
-سهولة التركيب.  
----------------  
2- عن طريق الnpm: بكتابة الأمر (npm i bootstrap):  
خطوات التركيب بعد التثبيت:  
-تدخل لملف الAngular json وتضيف فيه سطرين:  
الاول: تكتب "node\_modules/bootstrap/dist/css/bootstrap.min.css" فى الstyles.  
التاني: تكتب "node\_modules/bootstrap/dist/js/bootstrap.bundle.js" فى الscripts.  
(ملف الbundle؛ عشان الdropdown تشتغل معاك).  
عيوب الطريقة:  
-لا يفضل إستخدام js مع الAngular.  
(ملحوظة: ممكن تضيف ملف الcss الخاص بالبوتستراب فى الstyles.scss لو البروجيكت معمول scss عن طري ال@import.  
مميزات الطريقة:  
-الملفات بتكون ثابتة، وده هيخلي البروجيكت معتمد على نفسه، ومفيش هم مشاكل خارجية.  
-----------------  
3- عن طريق الngbootstrap:  
وده option وفرته الAngular بديل للbootstrap؛ لأن قديمًا كان البوتستراب معتمد على الjQuery وحاليًا على الjs، والاتنين دول غير مفضل إستخدامه مع Angular؛ فقدمولك الحل الجميل ده.  
خطوات التثبيت والتركيب:  
-اكتب أمر ng add @ng-bootstrap/ng-bootstrap في التيرمنال.  
وبس، هو هيعمل Update للملفات بنفسه.  
عيوب الطريقة:  
-لا يوجد به كل الComponents الخاصة بBootstrap.  
مميزات الطريقة:  
-يعتمد على الTypeScript بدل الjs وهو المفضل للAngular.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

لو سألت حد من الي بيشتغلوا Angular 14 بنفس الطريقة القديمة:  
-ايه الفرق بينها وبين 13؟  
هيقولك ولا حاجة.  
لكن فى الحقيقة تعالى اوضحلك جزء بسيط من العظمة الي بتعملها Angular.  
الAngular 14 عبارة عن Feature preview لما قادم فى الAngular، يعني كdeveloper غير ملزم انك تتعامل بالFeatues دي لحد ما تستقر، إضافات ومميزات كتير هتكلم عن ميزة واحدة بس منهم فى المقال ده (standalone components).  
الAngular 14 ركزت على مبدأ الMicro Frontend، ووصلت للهدف ده عن طريق خاصية الStandalone الي للوهلة الأولى هتحس أنها موجودة فى كل الFrameworks التانية، لو حسيت كدة تبقى متسرع.  
من الاخر لو جيت ومسكت الAngular وقسمتها اجزاء هيطلعلك اجزاء كتير جدًا:  
(Component, pipe, directives, services, guard).  
فتخيل بالفعل ان ده الي عايزه تعمله Angular 😃، وصلت؟  
لو لسه فده معناه انك نجحت فعلًا فى الوصول للMicro frontend؛ لأنك حرفيًا جزأت كل جزء فى اكتر فريمورك مكون من اجزاء.  
هتقولي هسألك سؤال:  
-بس الي انا اعرفه ان الAngular بتعتمد على الModules، يعني الكلام ده مش هيحصل.  
=هو انا مقولتلكش؟ الAngular فى نظام الStandalone خلت التلاتة الحلوين دول: (Component, directive, pipe) يتعاملوا معاملة الmodules تمامًا يعني يتعمل فيهم import ويتعملهم import عادي جدًا، معنى كدة ايه؟ ايوة الله ينور عليك (سرعة أكبر، ووصول مباشر).  
سؤال تاني معلش:  
-مش كدة برضو انت اتخليت عن الLazy loading، الي اعرفه ان الLazy loading بيشتغل على الmodules.  
=فاكر الLoadChildren؟ الي كانت بتload من module، عملوا حاجة شبهها اسمها LoadComponent، وبتعمل load للStandalone Component؛ فمعنى كدة انك هستتفيد من سرعة الوصول وكمان من الlazy loading، وده غير انك تقدر تعمل bootstrap للبروجيكت من خلال Standalone component، واي حاجة Standalone؛ فهي شبه معزولة عن باقي الابليكشن من حيث تأثير الmodules، يعني لو انت مثلًا مثبت angular material فى الappModule؛ فالstandalone مش هيسمع بيها؛ لأنه مش declartion؛ فلو حابب تضيفله حاجة فبتضيفهاله فى الimports بتاعت الcomponent نفسها.  
----------------  
ودي يا باشا ميزة واحدة من المميزات الي ضافتها Angular في سبيل الmicrofrontend، الموضوع عميق جدًا.  
حتى الي شرحت من داخل تيم Angular، قالت ان الميزة لسه قدامها شوية عشان تبدأ تبان قوتها، مع التعامل الكتير معاها هتبان فروق السرعة وجوانب قوية آخرى، وانهم لسه مش مستعدين لإتمام الفكرة تمامًا لازم ياخدوا فيدباك من الdevelopers ويعالجوا سوا المشاكل الي تظهر بسبب النظام ده لو حد حب يجربه.  
وغير انها وعدت ان نفس الكلام ده هيتعمم فى باقي الAngular؛ فمن الأخر انت على موعد جديد من الAngular والي هتكون عبارة عن 2 فريمورك إن جاز التعبير، فريمورك طبيعي الي احنا لسه شغالين بيه، وفريمورك مايكرو.  
ولعلمك الAngular 14 هي فقط Feature preview للميزات الي إضافت، يعني المهم هي التحديثات القادمة، لكن لو حبيت تجرب من دلوقتي جرب؛ لأن زي ما قولتلك لو هتشتغل على النظام القديم فأنت زي ما أنت كأنك بتشتغل 13 عادي جدًا.

-ما هو الview encapsulation فى الAngular وبيعمل encapsulate لايه وازاي وانواعه؟  
-ما هو التغليف فى الAngular وبيغلف ايه وازاي وانواعه؟  
(من أسهل اسئلة الانترفيو بس يبان صعب، وموجود فى اغلب الانترفيوهات).  
التغليف او view encapsulation ده عبارة عن specific attribute بيتضاف؛ عشان لو هتعمل style لحاجة معينة؛ بحيث انها متأثرش ف مكان تاني يعني مثلًا لو انت عندك component وفيها عنصر او class فأديته Style مينفعش تلاقي مثلًا نفس العنصر فى component تانية واخدة نفس الStyle؛ فالAngular تلقائي بتعمل عملية التغليف فمثلًا بدل ما العنصر يبقى h3 فى الstyles يبقى h3[\_ngcontent\_yax-c5]، ونفس التغليف يكون على عنصر الHtml بالشكل ده h3 \_ngcontent\_yax-c5>؛ فبيخليه معزول ومش بيشارك الstyle بتاعه مع حد.  
-----------  
والview encapsulation له 3 انواع:  
1- وده الdefault والي اتشرح فوق(Emulated):  
شوف شرحه فوق + طريقة كتابته بتكون ف الdecorator الي اسمه @Component بتضيف property او metadata:  
encapsulation: ViewEncapsulation.Emulated  
----  
2- الNone:  
وده بيتكتب كدة:  
encapsulation: ViewEncapsulation.None  
وده معناه انك متعملش تغليف لأي حاجة فى الComponent؛ فمعنى كدة لو عندك مثلًا h3 مديه لون؛ فأي h3 هياخد اللون ده كdefault طالما مفيش تأثير عليه من الcompnent التانية.  
----  
3- ال ShadowDom:  
وبيتكتب كدة:  
encapsulation: ViewEncapsulation. ShadowDom  
بيعمل ShadowRoot، وده بيغلف الcomponent بالكامل؛ فبالتالي مش بتتأثر بالglobal style، يعني مثلًا لو عندك style فى الملف الgloabal (style.css)؛ فمش هيأثر فيه ولكن لو في component تانية الViewEncapsulation بتاعها بNone؛ فهتأثر فيها عادي.

ايه الفرق بين الAhead Of Time(AOT) والJust In Time(JIT)؟  
(سؤال انترفيو)  
-هما الاتنين عبارة عن compiler بيحولوا الكود من شكل لشكل او من حالة لحالة.  
المقارنة:  
1- الcompile:  
الAOT: بتعمل compile وقت بناء البروجيكت داخل الIDE، مش بتحتاج انها تعمل compile فى الruntime.  
الJIT: بتعمل compile لما البروجيكت يروح للbrowser قبل ما يتعرض، وبتحول الTypeScript لjavaScript؛ لأن الbrowser مبيفهمش الTypeScript.  
----  
2- السرعة:  
الAOT: اسرع؛ لأنه بيعمل compile وقت كتابة الكود؛ فمش بيحتاج وقت.  
الJIT: ابطأ؛ لأنه بيعمل compile وقت الruntime.  
----  
3- الmode المفضل:  
الAOT: يفضل في الProduction mode.  
الJIT: يفضل في الdevelopment mode.  
----  
4- الcommand to run:  
الAOT:  
ng build --aot OR ng serve --aot  
الJIT:  
ng build OR ng serve  
----  
5- اكتشاف الاخطاء:  
الAOT:  
-بمجرد الحفظ، داخل الIDE؛ وبيوقف البروجيكت لما يلاقي خطأ ويطلعلك ايرور فى الCommand line والIDE والbrowser.  
الJIT:  
-عند استخدام الخطأ فى الbrowser، يكشف الخطأ فى الconsole.  
======  
الAngular بالوضع الإفتراضي يستخدم الاثنين:  
الAot بيعمل compile للكود داخل الIDE وبعدين يبعته للJIT يعمله compile ويحول الTypeScript لjavaScript.  
ممكن تعمل للAot داخل ملف الAngular json إيقاف، ووقتها لو عندك ايرور فى البروجيكت هيسيب البروجيكت يشتغل عادي، لكن لما تستعمل حاجة مش شغالة هيطلعلك ايرور فى الconsole.

ما هو الlazy loading فى الAngular؟  
(سؤال انترفيو)  
1- المصطلح:  
الLazy loading هو طريقة (Pattern)، بتستخدمه الAngular والعديد من الFrameworks الأخرى وأغلب لغات البرمجة، الغرض منه هو زيادة سرعة الApplication والوصول لperformance أفضل.  
----  
2- مع الAngular:  
الAngular بتستخدمه بطريقة ما للحد من عدد الmodules الي بتتحمل لما الuser يحتاجها، بالتالي الابليكشن يكون اسرع والperformance أسرع والIntial bundle size يكون أقل؛ فميحصلش بطأ لما تفتح الابليكشن.  
----  
3- الEger loading:  
والlazy loading على عكس نظام الEger loading تمامًا الي بيحمل كل الmodules الEger بمجرد دخولك للابليكشن؛ عشان كدة الlazy loading جه حل المشكلة دي بأنه يحمل الmodules الlazy عند طلبها.  
----  
4- طريقة عمله (ككود):  
بيستخدم loadChildren، والloadChildren بتستخدم توقع يرجع Function فيها Promise؛ بحيث انه يحمل الmodule فقط عند الدخول على component من الcomponents الخاصة بيه، الكود للتوضيح:  
{ path: 'profile', loadChildren: () => import('./profile/profile.module').then(m => m.ProfileModule) },  
والProfileModule: بيحتوى على الcomponents الخاصة به مستدعاه بداخله وبداخل الRouter module الخاص بيه.  
----  
5- التأكد من عمل الlazy loading:  
عند فتح الapplication، لن تجد اي من محتويات الcomponents التابعة للlazy module وجود فى الnetwork الخاصة بك، لكن عند الدخول الى الpath الخاص بالlazy module؛ فمثلًا في مثالنا عند الدخول إلى:  
yourPort/profile؛ ستجد ان الملفات الخاصة بالmodule تسارع ف الظهور.  
----  
6- لم تصلك الفكرة؟  
لديك الحق؛ لأننا هنا نتحدث عن module واحد فقط، لكن تخيل ان لديك ابليكشن يتسع لاكثر من 20 module او حتى 4 module؛ فبالتأكيد ستكون بحاجة ماسة، لمثل هذا الحل الرائع؛ فبتحويل جميع الmodules إلى lazy؛ سيكون الأمر أكثر سرعة وأقل حجمًا في الinitial bundle size.

ما الفرق بين الObservable والsubject؟  
(سؤال انترفيو)  
1- الاختلاف الرئيسي:  
الobservable:  
بياخد observer وبيتعامل معاه كأنه هو، يعني مثلًا:  
let myObservable = new Observable<any>(observer => {  
[observer.next](http://observer.next/)('Something');  
});  
الsubject:  
بيتعامل بشكل مباشر يعني مثلًا:  
let mySubject = new Subject();  
[mySubject.next](http://mysubject.next/)("something");  
----------  
2- الاختلاف الثاني (Hot vs cold):  
الObservable هو من النوع cold:  
يعني لازم يتعمله subscribe؛ عشان كل الي اتكتب فيه قبل كدة يشغله الابليكشن، مثال:  
let observable = Observable.create(observer => {  
[observer.next](http://observer.next/)('first');  
[observer.next](http://observer.next/)('second');  
...  
});  
[observable.map](http://observable.map/)(x => ...).filter(x => ...).subscribe(x => ...)  
لاحظ الsubscribe جات فى الاخر.  
--  
الSubject من نوع hot؛ فهو المطلوب انك تعمل subscribe فى الاول، واي جديد هيتضافله عادي، مثال:  
let source = new Subject();  
[source.map](http://source.map/)(x => ...).filter(x => ...).subscribe(x => ...)  
[source.next](http://source.next/)('first')  
[source.next](http://source.next/)('second')  
لاحظ انك ضيفت للSubject عن طريق الnext بعد ما اتعمله subscribe عادي.  
---------  
3- الاختلاف الثالث (UNICAST VS MULTICAST):  
الObservable:  
بتكون Unicast، يعني ايه؟  
يعني مثلًا لو انت عندك متغيرين وعايز تبعتلهم ارقام عشوائية؛ فلما تعمل subscribe لكل واحدة بحيث انه يغير قيمة الاتنين؛ فهو بيرجع قيمة مختلفة لكل متغير، مثلًا: 9|5.  
الSubject:  
على العكس تمامًا بيكون multitask:  
يعني لو معاك متغيرات كتير هيرجعلهم هما الاتنين نفس القيمة مثلًا:  
8|8|8|8  
لحد هنا تمام، بس لو عايز تعرف السبب؛ فده لأن برضو الSubject بيكون hot وبيعمل عملية الnext بعد الsubscribe، على عكس الobservable بيبعت الداتا قبل ما يعمل subscribe.  
-----------  
في النهاية بشكر البشمهندس Hussein Abd Elaziz؛ لأن الفيديو بتاعه ساعدني كتير فى المنشور ده وكذلك في المنشور السابق المقارنة بين الJIT والAOT، ورابط الفيديو فى اول كومنت.

ما هو الAngular interceptor؟  
(سؤال انترفيو).  
قبل ما اقولك ايه هو الinterceptor ومعناه العربي (المعترض)، خليني اذكرك بمعلومة ان الfrontend عندنا مجرد Ui؛ فهو لازم يعمل عمليات تخليه يتواصل مع الbackend؛ عشان يقدر يعرض الداتا والعمليات دي كلها تتلخص فى عمليتين (Request و Response)، الRequest هو ان الFront او الUi يطلب من الbackend الداتا، الResponse هو ان الbackend يجاوب على الطلب بتاع الfrontend ويبعتله المطلوب.  
----  
فالAngular قالت يعني ايه يتواصلوا مع بعض كدة على طول؟ احنا Framework محترم مفيش ظابط مفيش رابط؟ راحوا جابوا الInterceptor او المعترض، وده يا سيدي قاطع الطريق فى الAngular، رضيته هتعدي وتكمل مهمتك، مرضيتهوش وحصلت مشاكل، طلبك هيترفض (الRequest).  
----  
نجاوب سؤال الانترفيو بقى:  
لو المنصب الي مقدم عليه جونيور؛ فممكن تقول التعريف فقط، انه:  
بيكون medium بين الbackend والfrontend؛ فعند عمل request بيعترضه ويتعامل معاه ويبعته للbackend ويرجع بالresponse.  
لكن لو المنصب أعلى لازم تذكر شوية أمثلة حلوين، ولازم تبقى عارف ان الinterceptor مش بيهندل او بيعمل التواصل بنفسه؛ هو بس بيعمله perform؛ عشان يعمل المطلوب منه بالظبط.  
----  
تعالى نشوف إستخدامات صاحبنا ده، هو له العديد من الإستخدامات، بس أشهرهم 10:  
1- الloaders: الي بيحمل قبل الصفحة.  
--  
2- تعمل Changing the URLs.  
--  
3- تعمل check للErrors؛ عشان يعرف الAPI شغالة ولا لا (يعني نفس طريقة الAjax) الي هي 400 و500 و200، هارش؟  
--  
4- تعمل Notifications ويبعتهالك فى الconsole ولا بأي طريقة انت تقوله يبعتهالك؛ عشان مثلًا يقولك يا باشا الApi اشتغلت ولا الApi مش لاقيها.  
--  
5- تعمل Fake backend، يعني انت مثلًا معندكش backend تفضل كدة من غير ما تجرب الconnecting والكود بتاعك؟ يرضيك ده مثلًا؟ فأنت تقدر تعمل جواه Fake backend وتعمل تيست براحتك للصبح.  
--  
6- الAuthentication: تعمل check من خلاله على الtoken على الUser، عيش براحتك.  
--  
7- تعمل Headers.  
--  
8- تقدر تعمل بيه Converting من حاجة لحاجة، مثلًا تعمل convert لملف XML وتخليه json.  
--  
9- تقدر تعمل الProfiling: يطلع ايه ده؟ ركز معايا عشان ده مهم، الProfiling من الاخر يعني تزامن للعمليات، يعني لو انت عملت اتنين request ورا بعض او سجلت بأكونتين لنفس الdatabase؛ فهو بسبب التزامن مش بيتأخر في انه يعرف ان دول طلبين لنفس الحاجة مش طلبين مختلفين، من الاخر (without any time delay).  
--  
10- تعمل Catching: وده مختلف شوية عن الcheck errors؛ لأنه لما بتعمل Catching فأنت بتتأكد من عملية الGet نفسها وعلى أساسها بتهندل بالoperators المتاحة بتاعت الRxjs.

ما هو الcomponent فى الAngular؟  
--  
1- التعريف:  
الComponent هو اللبنة الأساسية لواجهة المستخدم (UI) في تطبيق الAngular، يحتوي تطبيق الAngular على شجرة (Tree) من الComponents.  
--  
2- النوع:  
هي مجموعة فرعية من الDerctive، يعني نوع خاص منها؛ لأنه دايمًا مرتبط بقالب، والنوع ده فى الDirective اسمه Component directive، (هتلاقي عندي مقال بيشرح الdirectives بأنواعها).  
--  
3- متطلبات:  
الComponent العادية لازم تنتمي لNgModule؛ عشان يكون مسموح لComponent تاني انه يستخدمه؛ فلازم يكون موجود فى الdeclarations بتاعت Module، والنوع الغير عادي هو الStandalone component والنوع ده ظهر فى Angular 14 كFeature preview، (هتلاقي عندي مقال بيشرح الStandalone component).  
--  
4- الLife cycle:  
وطبعًا لإن الComponent هو فى الأصل Directive؛ فهو له Lifecycle Hooks وConstructor الي بيتضاف فيها الServices بتاعتك، (هتلاقي عندي مقال بيشرح الLife cycle).  
--  
5- مكونات الComponent:  
بيتكون من 4 ملفات (ممكن تختصرهم فى ملف واحد):  
1- الTs file وده اهم ملف فى الComponent، وده لإنه بيتكتب فيه الdecorator الخاص بيه @Component، وبداخله بيتكتب شوية meta data بتوضح اسم الComponent والملفات الباقية المعتمد عليها، وكمان هو الController بتاع الComponent يتكتب بداخله الFunctions والLife cycle والConstructor الي بيتضافلها الServices المساعدة، وفي اشياء اكثر زي الchangeDetection وغيرها لكن المقال مش هيكفي لذكر الكل.  
2- الHTML file:  
وده الView او الTemplate، وبيتضافله كود الHTML والDirectives المساعدة زي الngFor وngIf، ويتعمل فيه Binding للdata الموجودة فى الTs file غير طبعًا الPipes، والملف ده بيتعرف داخل الTs file بtemplateUrl، وبيتحط الpath بتاعه او يمكن الاستغناء عن الملف بالكامل وتعمله داخل الts عن طريق كلمة template، وده لا يفضل إلا في حالة ان الcomponent صغيرة، بحيث انك تضيف حاجتك داخلها.  
3- الStyle file:  
بيكون موجود فيه كود Css او Scss او Sass او Less حسب ما انت اخترت تشتغل بيه، ومش محتاج انك تعمل Compile لأي نوع؛ لأن الFramework بيعمل الcompile بنفسه، والملف ده بيتعرف داخل الTs file بstyleUrls وخلي بالك انك تقدر تضيف اكتر من ملف Style، وبيتحط الpath بتاعه او يمكن الاستغناء عن الملف بالكامل زي الHtml بس هيبقى style مش Template.  
4- ملف الTest:  
وده بيكون ملف Test داخل الAngular وبيشتغل بkarma وJasmin فى الوضع الإفتراضي.  
----  
لحد هنا الكلام مخلصش بخصوص الAngular component، يعني المقال ده فكرة توضيحية سريعة عنه.

ما هي الPipes فى الAngular؟  
(المقال يتضمن سؤال انترفيو)  
1- التعريف:  
الPipes يعني انابيب ومن الاخر دي مسؤولة عن انها تعمل Format للdata داخل التمبلت بمساعدة الtemplate binding، وهي بتحول الdata من صورة داتا عادية للصورة الي احنا عايزينها، من الاخر دخول الpipe مش زي خروجه.  
---  
2- طريقة الكتابة:  
زي ما قولنا فى التعريف بمساعدة الtemplate binding؛ فبتكون بالشكل ده:  
{{ data | myPipe }}  
---  
3- بعض المصطلحات وشرحها:  
-الbuilt-in pipes:  
دي الpipes الموجودة فى الAngular (مش بتدخل في بناءها)؛ وهتلاقيها كلها فى الDocumentation بتاع الAngular من اللينك ده: <https://lnkd.in/dnEqiVwZ>  
--  
-الcustom pipe:  
دي الpipe الي بتبنيها بنفسك من الصفر وبتعمل عملية مش موجودة فى الbuilt in pipes مثلًا لو حابب تعمل pipe للخصم او كدة.  
--  
-الimpure pipes:  
الpipes دي شرحتها فى مقال قبل كدة مقارنة بينها وبين الpure pipes، بس كشرح سريع لفكرتها؛ فهي Pipes غير متزامنة (Async)، يعني بتأدي دورها بعد ما كل العمليات المتزامنة على الdata ما تخلص، وفي منها built-in pipes وcustom pipes، المقال هتلاقي فيه شرح اكتر.  
--  
-Parameterized Pipes:  
بإختصار دي pipes بتاخد params، مثال:  
{{ today | date:’fullDate’ }}.  
--  
-Chaining Pipes:  
وده مصطلح معناه انك تقدر تعمل اكتر من pipe لنفس الحاجة، مثلًا:  
{{ today | date:’fullDate’ | uppercase}}.

ما هو الAgile؟  
-التعريف:  
الAgile هي منهجية لتطوير البرمجيات، تنبع من فكرة تقسيم كميات كبيرة من العمل إلى اجزاء صغيرة، يمنح هذا مديري المنتجات والمطورين وأي صاحب مصلحة فهمًا أفضل للعمل المراد ويزيد من قيمته، يختص في بناء المشاريع المبهمة والتي يوجد بها مخاطر.  
(الAgile هي Mindset).  
-سبب وجود المنهجية:  
قديمًا، كان تطوير البرمجيات عملية بطيئة؛ حيث يمكن للتغييرات الرئيسية في المتطلبات أن تضع ضغوطًا ومخاطر كبيرة على الفرق.  
-الحل الذي قدمته Agile:  
عند اتباع منهجية Agile، تساعد الأجزاء الأصغر من العمل الفرق على أن تصبح أكثر مرونة؛ فهي تساعدهم في هذه العملية على تقديم الميزات بشكل أسرع والاستجابة للتغييرات بشكل أسرع.  
-----  
أفكار الAgile اتقسمت لاكتر من نوع من أطر العمل (frameworks)، لكن أشهرهم اتنين: Scrum وKanban.  
وبإختصار عمل أطر العمل بيعتمد على مهام متكررة مبنية على منهجية الAgile.  
-----  
وبشرح سريع لواحد منهم هتفهم عملية الAgile بتم ازاي؛ فتعالى نشوف الScrum بيشتغل ازاي، وقبله لازم تكون عارف مصطلح الSprint.  
الSprint:  
فترة زمنية قصيرة ومحددة زمنيًا عندما يعمل الفريق لإكمال قدر معين من العمل،وهي صميم منهجيات Scrum ورشيقة، وبتساعد الSprints الصحيحة فريقك المرن على شحن برامج أفضل مع عدد أقل من الصداع.  
ولازم طبعًا الاول تتأكد إن الفريق محدد وفاهم هدف الsprint وازاي يقيس نجاحه، ده هو الاساس للحفاظ على توافق الجميع والمضي قدمًا نحو وجهة مشتركة.  
-----  
الScrum:  
هو إطار عمل يمكن للناس من خلاله معالجة مشاكل التكيف المعقدة، مع تقديم منتجات ذات أعلى قيمة ممكنة بشكل منتج وخلاق.  
وبيخضع لأربع عمليات رئيسية:  
1- الProduct owner يبعت طلب التحسينات في صورة Backlog.  
2- الTeam ياخد الBacklog ويحولها لSprints او يرجعها فى صورة Sprints.  
3- يقوم الTeam وصاحب المصلحة بفحص النتائج ويعرفوا اذا كانت متوافقة مع بعضها او لا، وبعدها يتعمل Sprints تانية.  
4- تعاد العملية لحين انتهاء العمل بالكامل.  
والعملية بأكملها بتخضع لScrum master بيراجع العملية وبيشرح للكل دوره والقواعد الي يمشي عليها، وهو الSoftware leader لTeam الScrum.  
وطبعًا بعد ده كله يخضع البروجيكت لتجربة المستخدمين والشكاوى ويتم تطوير البروجيكت بناء عليه.  
وده Framework من الFrameworks المطبقة لنظرية الAgile.  
-----  
طبعًا المقال مش أحسن حاجة لشرح عملية الAgile، لازم تدخل فيها بشكل عملي؛ عشان تفهمها وتتعمق فيها أكتر وتتعود عليها، غير انك ضروري بالتأكيد هتحتاج لكورس يشرح العملية بشكل أفضل.

ما هي الAngular building blocks؟  
(يتضمن العديد من اسئلة الانترفيو)  
المقال ده كوكتيل، بديك ملخص سريع عن دور الbuilding blocks فى الAngular، وخلي بالك ده دورهم مش تعريفهم، التعريف حاجة تانية وهعمل مقال منفصل عنه بإذن الله.  
----  
الbuilding blocks:  
يعني الblocks المكونة للبروجيكت بتاعنا، لو هنفترض ان البروجيكت عباراة عن ماتش كورة فكل واحد من الbuilding blocks بيأدي دور مختلف عن التاني يساعد في تكوين البروجيكت، وفى المقال ده جمعتلك اهم الblocks المهمة لعمل بروجيكت Angular.  
----  
1- الModules:  
الmodule هو كتلة من التعليمات البرمجية، لها مسار عمل واحد ومحدد، اي تطبيق Angular لازم يكون في module واحد على الاقل وهو الroot module، بس أغلب البروجيكتس ليها اكتر من module.  
--  
2- الComponents:  
هو الbuilding block الرئيسي ومش بس فى الAngular، لا في اي Frontend Frameworks؛ لإن كل الFrontend Frameworks انشأت لبناء SPA appliction، والي بيخدم العملية دي هو الComponent، ومهمته بتكون هي العرض (View)؛ بحيث ان كل عرض بيعرضه بيكون جزء من الشاشة، وطبعًا اي تطبيق Angular لازم يكون فيه على الاقل component وهو الapp root وبيتعمله bootstrap فى الroot module، الComponent قبل كدة شرحت مكوناته وبإختصار هما تلاتة مهمين فى عملية الdev:  
الclass وده بيكون فيه كل العمليات.  
الView وده بيكون التمبلت الUi.  
الStyle وبيكون فيه اكواد الStyle.  
--  
3- الData binding:  
نعم، هو جزء من الComponent لكنه Building block؛ لأنه من أساسيات الAngular ان يحصل Data binding بين الTs والHtml، والblock ده معناه الربط بينهم مش اكتر.  
--  
4- الDirectives:  
بتعدل سلوك الDom بالإعتماد على طبيعتها الديناميكية وتم شرحها في مقال مسبق، وليها اكثر من نوع، يفضل تشوف المقال بنفسك: <https://lnkd.in/dNUHZ6Zx>  
--  
5- الMetadata:  
الClasses بيكون ليها Metadata يعني بيانات وصفية توضح الغرض منها؛ عشان تساعد الDecorator في فهم الغرض من الClass، لإتمام عملية الProcessing.  
--  
6- الDependency injection:  
ده عبارة عن Design pattern بتستخدمه الAngular عشان يقدملك الdependencies الضرورية للإستخدام مع الComponent؛ فمثلًا بتستدعي فى الConstructor بتاع الcomponent بتاعتك الServices والModules المطلوبة، زي كدة مثلًا:  
constructor(private router:Router, private myService: MyService)  
وبالشكل ده تقدر تستخدم istance من الmodule او الservice داخل الclass بالكامل، يعني داخل وخارج الconstructor، لكن لو من غير كلمة private يبقى داخل الconstructor بس.  
--  
7- الServices:  
بستغرب من الخلط الكبير الي بيحصل بين وبين الDependency injection؛ وده لأنه بيشتغل فى الcomponent بالDI، بس الServices عبارة عن شوية Functions بتلعب دور مهم فى الApplication وتقدر تقدر تاخد instance منها عن طريق الDi، واكثرية تعامل الServices بيكون مع الAPIs.  
--  
8- الRouting:  
وده Topic كبير جدًا ممكن تختصره فى جملة (بيختصر روابط الUrl لشكل انت تحدده)، بس طبعًا له حوارات وتفاصيل، واتكلمت عن شوية تفاصيل فى الاكونت بتاعي.  
--  
9- الTemplate:  
هو جزء من الComponent ومرتبط بالcomponent دايمًا، وبيساعدك في بناء الUi الخاص بيه، وبتستخدم بداخله الDirectives والPipe لمعالجة سلوك الDom والبيانات (Data) بداخله.

ازاي تعمل Share لdata بين اتنين components او directives فى الAngular؟  
(سؤال انترفيو مهم).  
السؤال ده من الاسئلة الي بتوضح الفهم؛ فلازم تركز وتبقى عارف تبدأ الإجابة منين وبأحسن شكل.  
-----  
اول حاجة لازم تعرف مفهوم الParent والChild فى الAngular، وهو ان التمبلت بتاع الparent بيكون جزء من مكوناته الchild، يعني افترض ان الكود ده داخل الparent:  
<h1>  
I am parent component  
</h1>  
<app-child></app-child>  
-----  
الإجابة بشكل مباشر:  
-الshare من الparent للchild:  
1- له طريقة واحدة وهي بال@Input وده property decorator.  
---  
-الshare من الchild للparent وده له طريقتين:  
2- الأولى: وهي من خلال ال@Output وده property decorator مع طبعًا الEventEmitter.  
3- التانية: عن طريق ال@ViewChid وده برضو property decorator.  
---  
4- الShare بين اتنين component ملهمش علاقة ببعض:  
وده بيتم عن طريق الServices.  
---  
والأربعة دول هم إجابة السؤال، وإن شاء الله إجابة كافية، لكن لازم تكون اتدربت عليهم؛ فأنصحك تشوف Tutorials عنهم لإن ممكن تتسأل فى شوية تفاصيل أكتر.  
وطبعًا ممكن تعمل share للdata عن طريق الstate management زي الngrx بس ده حوار لوحده.

بسم الله الرحمن الرحيم.  
  
ما هو الMulti-router فى الAngular؟  
ازاي تعمل اكتر من router-outlet فى نفس الcomponent وشكله بيكون ازاي ومنهم واحد بيعرض components مختلفة بس داخل الcomponent الي بيعرضها الاول؟  
(ده واحد من أهم اسئلة الانترفيو ل2022، بل وبيكون فاصل فى القبول، رغم انه ان شاء الله هيكون سهل، لما نجاوبه سوا).  
----  
1-الPrimary router outlet:  
تعالى معايا فى جولة بسيطة داخل الrouter-outlet يعني (behind the scenes)، الrouter outlet فى الوضع الإفتراضي بيتخلقله name الname ده بيكون اسمه primary.  
ايه لازمة المعلومة دي؟ هقولك.  
----  
2- الSecondary router outlet:  
بيكون router outlet بس انت بتحدد الاسم بتاعه، بالطريقة دي:  
<router-outlet name="sidebar"></router-outlet>  
وتيجي داخل الrouting module بتاعك وتضيفه بالشكل ده للcomponents الي عايزه يعرضها:  
{ path: 'chat', component: ChatComponent, outlet: 'sidebar' }  
لاحظت كلمة outlet؛ اهي دي بتقوله ان الrouter-outlet الي تسمعله هو الي اسمه sidebar.  
----  
3- شكل اللينك Url عشان توصل لcomponent من الSecondary:  
localhost:4200/home(sidebar:chat)  
تعالى نفصصه:  
الlocalhost بالأرقام 4200 ده الdomain الي انت شغال عليه.  
الhome (دي الcomponent) الي فيها الSecondary.  
لاحظت حاجة غريبة صح؟ (sidebar:chat)، دي بقى عشان ده Secondary router؛ او انه router معمول custom للإسم بتاعه؛ فلازم يظهر كدة.  
----  
4- طريقة كتابته كrouter-link:  
طبعًا انت كواحد عامل path؛ فلازم تعمله button بيودي عليه، وبيتكتب بالطريقة دي مع الbutton:  
<a [routerLink]="[{ outlets: { primary: 'home', sidebar: 'chat' } }]">Chat</a>  
لاحظ انه اداله الprimary بتاعه، والSecondary، وطبعًا هتلاقي اكتر من شكل لكتابة الrouterLink، بس كتبتلك الأفضل بالنسبالي.

الAngular بيتم إستخدامه على نطاق كبير في تطوير الويب، وهو أهم Framework خاص بالFront end، بيحظى بدعم جوجل، وله architectures متنوعة زي الMVC والMVVM، وكمان له العديد من الCore modules الخاصة بيه غير الRxjs الي بتمثل الReactive programming والTypeScript والي هي Strong Typed language؛ وبما إننا اتعرفنا على العملاق ده في مقالات سابقة، انا جاي النهاردة اقولك المتطلبات قبل البدء فيه.  
----  
1- الHTML, CSS, javaScript:  
لازم يكون عندك خبرة جيدة بيهم، وتكون نفذت بعض المشاريع الي بتضمن التعامل مع localstorage او backend، كمان تكون اتعاملت مع Cruds وWeb api وAjax & json، يفضل تكون كمان اتعاملت مع Framework css زي الBootstrap او Tailwind، كذلك لازم تكون عارف امتى وازاي تستخدم Es2015 الي هي (ES06)؛ لأن التحديثات الخاصة بالإصدار ده مطلوبة وقت التعامل مع الAngular زي الarrow function والClasses، والModules وغيرها من المميزات الي هتكون ضرورية.  
--  
2- فكرة بسيطة عن الMvc:  
وده architecture pttern بيتعامل مع الModel والView والController وقريب جدًا هيكونله مقال منفصل (او مقالات).  
--  
3- معرفة بسيطة فى التعامل مع الcommand line:  
وده لإن الAngular بيوفرلك Cli والاسم ده إختصار لجملة (Command line interface)؛ وده بيوفر عليك وقت طويل في بناء العديد من الbuilding blocks الي هتستخدمها فى بناء تطبيق الAngular.  
--  
4- معرفة بسيطة بالNpm:  
اوقات بتحتاج لتثبيت او إضافة بعض الpackages لمساعدتك في بناء تطبيق الAngular؛ فبالتأكيد هيكون اختيار ممتاز انك تختار Npm، طبعًا هتحتاج تثبت Node js؛ وانت بتثبته ثبت النسخة المستقرة مش احدث نسخة؛ عشان ميحصلش مشاكل.  
--  
5- دراسة الTypeScript:  
وهي لغة برمجة مش بتختلف عن الjavaScript كتير، بس بتكون Typed وبتساعدك اكتر في بناء التطبيق، بمرونة اكتر وبشكل منظم ومتكامل، ولو دخلت على الهاشتاج الخاص بيا هتلاقي شرح ليه هي أفضل من الjavaScript.  
--  
6- الRoadmap:  
دور على Roadmap كويسة، واعرف هتذاكر ايه ودور بعده على كورسات، قريب هشرح Roadmap وبعدها هرشحلك مصادر، خليك متابع الهاشتاج الخاص بيا، وان شاء الله تستفاد [#NgOmarRa](https://www.linkedin.com/feed/hashtag/?keywords=ngomarra&highlightedUpdateUrns=urn%3Ali%3Aactivity%3A6985156423743619072).

بسم الله الرحمن الرحيم.  
ما هو الApi؟  
الAPI:  
هي واجهة التطبيق البرمجية.  
بتعمل ايه؟  
-هي عبارة عن ميكانيزم لربط اتنين ابليكشن او اكتر، يعني مثلًا لو غيرت صورة او نشرت بوست من خلال الاكونت بتاعك في الفيسبوك من خلال الموبايل، الاكونت بتاعك فى الويب بيحصله نفس التغيير.  
-واجهة التطبيق البرمجية = عمليات التطبيق ككل؛ فهي بتحل محل التطبيق.  
----  
اول حاجة لازم ترجع بالزمن لورا، قبل ما يكون ضروري إيجاد حاجة زي الAPI، كان الويب او الموبايل عبارة عن تطبيق واحد backend وله UI وكان دور الjavaScript او لغات تصميم الموبايل، رغم الإمكانيات القوية مقتصرين على جانب الUI اكتر من الجانب البرمجي، وكان طبيعي وقتها ظهور مكتبة زي الjQuery الي احتلت الويب لفترة كبيرة جدًا ومازالت موجودة؛ فكان وقتها يعتبر الFrontend معني بالUi فقط فممكن تقول عنه Ui developer؛ لأن فى النهاية البروجيكت بيعمله الbackend.  
----  
فى العصور دي من الويب وقبل الوجود الكبير للموبايل زي دلوقتي، مكنش في حاجة اسمها نعمل 2 ابليكشن لنفس الdata؛ لإن الباك بالفرونت بالداتا بكل حاجة ف الموقع تطبيق واحد، صعب تجزئته، وكمان مكنش ضروري لان الوقت ده كان اغلب الانترنت عبارة عن مواقع ويب بس، ولما ظهرت تطبيقات الموبايل فى البداية مكنش في ضرورة لربطها بتطبيقات الويب.  
----  
بعد ظهور الAPI، الBackend انفصل عن الفرونت وبدل ما هو كان تطبيق واحد بقى تطبيقين، backend وFrontend.  
هتسألني تقولي ايه المميزات الي عادت عليا منه؟ تعالى اقولك:  
1- ده سمح لوجود اكتر من تطبيق لنفس الdata.  
2- الbackend يشتغل لوحده والfrontend يشتغل لوحده، وده ساعدك تطور في المجالين، ان كل مجال منهم مش مرتبط بالتاني؛ فمثلًا الفرونت اتوجد فيه بعد كدة مصطلحات زي الSPA والCross platform، والحاجات دي صعب تكون موجودة بدون API.  
--  
3- لما فصلت الاتنين عن بعض؛ فأنت خليت الbackend زي صندوق معزول والFrontend يقدر ياخد منه ويشكل لاكتر من شكل، بمعنى انك ممكن بنفس الAPI تعمل ابليكشن ويب وموبايل زي ما قولتلك في مثال الفيسبوك، واكتر من كدة بكتير، ممكن تطلع من الAPI عدد لا نهائي من التطبيقات، وكلهم يرجعوا لنفس الAPI ولنفس الdata.  
--  
4- الAPI مفهومة بالنسبة للbackend والfrontend؛ فهي مكتوبة بالJson، ودي طريقة كتابة برمجية سهلة جدًا موجودة ف الjavaScript

ما هي الفروق بين الInterfaces والModel classes؟  
(سؤال انترفيو)  
----  
بصراحة في المقال ده، انا جاي اقولك كبداية ازاي تبطل تستخدم any او تقلل إستخدامها بنسبة 90% وبعدين نبدأ في مقالنا، الApi الي انت هتتعامل معاها هتكون مبعوتالك Json، ازاي تحولها لInterface او Model؟  
بص في طريقتين والطريقتين ساهلين:  
الاولى انك تعملها يدوي، والتانية انك تدخل ع موقع Transform tools وتختار json to typescript وتديه json هيطلعلك Interface.  
----  
تعالى نبدأ في سؤال الانترفيو، الInterface مقابل الModel class:  
1- الإجابة البسيطة:  
الInterface والClass مشتركين في اني اعمل Typing للData بتاعتي، بس الفرق الInterface ف الوضع الافتراضي لما اطلبه عشان اتأكد من نوع الداتا، ارجعله كل الداتا المطلوبة، لكن الClass بيفرض اني اقوله المتغير ده مطلوب ولا optional، تعالى اضربلك مثالين:  
الInterface:  
export interface Root {  
id: number  
title: string  
}  
لازم ارجعله الاتنين والا مش هيشتغل او الCompiler هيوقف البروجيكت.  
  
لكن الModel Class:  
export class Root {  
id!: number  
title?: string  
}  
الاولى معناها لازم، التانية optional، يعني ممكن مترجعش data للمتغير ده.  
---  
2- الcompile والRun time:  
-الInterface:  
وقت الCompile بس بيكون موجود، ولا يعتبر جزء من الكود، بيسمحلك تتأكد بس من البيانات بتاعتك وان البنية صح، ولما يتأكد ان كل حاجة صح، يسمحلك ان البيانات توصل للواجهة؛ فهي اتعملت للراحة والسهولة، ومش بتاخد مساحة ولا بتكون موجودة وقت الRun time.  
-الClass:  
الكود بيفضل فى الRun time يعني مش بيتمسح، بيكون موجود ف الحالتين، الCompile والRun time؛ اتعملت للراحة والسهولة، بس بتاخد مساحة.  
---  
3- امتى استخدمها، وامتى لا؟:  
--الInterface:  
-تستخدمها لما تكون عايز شكل مكون للData المطلوبة فقط، وكلها تكون تعريفية للمطلوب، مش بتعمل عملية معينة، وخاصة لما تكون data هتستخدمها في اكتر من Component او Service.  
-متقربلهاش لو عايز يكون في default value للمتغير بتاعك، او انك عايز تعمل implement لاي حاجة، او انك محتاج الConstructor او محتاج انه يعملك function، يعني مثلًا لو الداتا الراجعة يتغير من شكلها تضيف كلمة او معادلة، إلخ.  
--الModel class:  
-تستخدمها لما تحب تطلع instance من الData الي رجعالك، او انك تعمل default value، بتدعم انك تستخدم الConstructor ووجود Functions.  
-لا يفضل استخدامها لو الData قليلة وملهاش default value، وخلي بالك ان الاتنين بيعملوا Type checking؛ فلو مش هتستخدم حاجة خاصة من خواص الClass، يفضل تستخدم الInterface.  
---  
4- نقاط إضافية:  
-الexport keyword:  
الInterface: بتاخد interface keyword.  
-الClass: بتاخد class keyword.  
-الReal time use:  
الInterface: مرتبط بالarchitecture.  
الClass: هو Design pattern مرتبط بالproject structure.

ما هو الData binding فى الAngular وأنواعه؟  
او اوصف الData flow وصنفها.  
(سؤال انترفيو)  
الData binding:  
-هو ربط بيانات ملف الTypeScript (ملف الSource) او ممكن تقول عنه عقل الComponent وملف الHtml (ملف الView)، الي بيتعرض فيه الUi الخاص بالComponent.  
---  
وبينقسم الData binding فى الAngular لثلاث أنواع رئيسية ضمن فئتين:  
فئة الOne way data binding:  
1- من الSource للView.  
2- من الView للSource.  
فئة الTwo way data binding:  
3- من الView للSource للView.  
-------  
1- من الSource للView:  
وده بينقسم ل3 انواع:  
-الInterpolation:  
وده بيعمل bind للdata بالشكل ده {{data}}.  
-الProperty binding:  
وده بيعمل bind لاي Property html بالشكل ده:  
<img [alt]="[hero.name](http://hero.name/)" [src]="heroImageUrl">  
لاحظ انه هنا عمل bind لalt وعمل bind لSrc الصورة.  
والعملية دي بتساعد في عملية الInput الي بدورها بتنقل الdata من الparent للchild.  
-ال[[bind.target](http://bind.target/)]:  
وده انت بتحدد نوع الbind وامتى تشتغل، مثال:  
[class.special]="isSpecial"  
وده معناه ضيفلي special لمجموعة الclasses الي عندي لو الisSpecial بTrue او موجودة.  
وممكن تعمل bind بنفس الطريقة لStyle اوAttribute.  
----  
2- من الview للsource:  
وده نوع واحد وهو الEvent Binding:  
ومعناه انك بتبعت طلب من الView للSource بتطلب منه Function، وله أمثلة كتير:  
زي ال(click)="onSave()"  
وده لما يضغط هيستدعي الfunction ال اسمها onSave، وممكن تبعت معاها param زي كدة (click)="onSave($event)".  
واي Function عمومًا ف الjavaScript ممكن تستخدمها زي الmouseover وزي الازرار (keydown.shift.t)="onKeydown($event)" ودي Function مش هتشتغل الا لما تضغط على زرارين الshift والt مع بعض.  
غير انها بتساعد في عملية الOutput الي بتنقل فيها الdata او الevent من الchild للparent.  
----  
3- من الView للSource للView:  
او الTwo way data binding وده بيعمل target لحاجتين الEvent والProperty؛ فهو يعتبر جزئين بالشكل ده:  
من الView للSource ده Event binding.  
ومن الSource للView ده Property binding.  
واشهر مثال عليه هو:  
ال<input [(ngModel)]="name">  
وده كنا بنستعمله بشكل أساسي ودايم مع الTemplate driven forms ولكن حاليًا في بدايل أفضل ومأمنه أكتر، ولكن برضو ما زال يستخدم ومفيش مشاكل مع إستخدامه، وده بيعمل تزامن للdata بين الview والsource.

ما هي الReusable components في الAngular، وأمثلة عليها؟  
(سؤال انترفيو)  
لما الAngular اتعمل، اتعمل لتعزيز إمكانية إستخدام الكود، معنى كدة انك تحاول تستفاد من الكود اكتر من مرة بدل ما تنشئ جديد؛ فالresuable components هي components قابلة للإستخدام اكتر من مرة وفي مختلف الصفحات، والعملية دي بتم عن طريق انك تعمل Child components لParent Component او page ومن خلال اكتر من Decorator، أشهرهم الInput والViewChild ودول Property decorators يعني بيتعاملوا مع الخواص.  
----  
الarchitecture المتبع عند بناء الAngular عند غالبية الأشخاص الي عندهم خبرة بيكون في Shared module ومن ضمن ملفات الShared ملف اسمه partials، او مش شرط يكون مع الShared بس ده مش موضوعنا.  
----  
ملف الpartials ده بيكون فيه Reusable components وcomponents زي الHeader والFooter.  
----  
زي ما قولتلك الResuable components هي قابلة للإستخدام اكثر من مرة؛ فهديك أمثلة عليها:  
1- الTitle:  
وده أبسط مثال فيهم.  
في مواقع كتير بتلاقي الTitle الي في الصفحة بيتغير، مثلًا Cart - Blog، إلخ...؛ فأنت ممكن تعمل Reusable components في الحتة دي بالشكل ده:  
هتيجي في الResuable وهي هنا Child component وتكتب الكود ده في الView:  
<h1>{{title}}</h1>  
والكود ده في الComponent Ts:  
@Input() title!: string;  
وهتيجي في الView عند الParent component، وتكتب الكود ده:  
<app-title title="Blog"></app-title>  
فتلقائي الجزء ده في صفحة الBlog هياخد كود الHtml والCss.  
---  
2- الCard.  
3- الButtons.  
وأمثلة كتير مش بتخلص وكمان عندك مكتبة زي الAngular material خير مثال على فلسفة الResuable components، والموضوع مش بينتهي عند نقطة مشاركة الHtml والCss بس، في كمان مشاركة الTypeScript، وهديك مثال:  
كنت شغال قبل كدة في بروجيكت Angular مع زميل؛ فكان مهم بالنسبالي ان يكون الكود ممتاز والشخص ده يكتسب خبرة، وكان عندنا 3 صفحات فيهم Reactive Forms؛ فلقيت اني ارشحله مقالة فيها الكود النهائي للInput والLabel بالشكل ده:  
<text-input  
[control]="[fc.email](http://fc.email/)"  
[showErrorsWhen]="isSubmitted"  
type="email"  
label="Email">  
</text-input>  
والكود ده كان بيطلعلك الErrors والInput له Label وType والControls المتحكمة بيه؛ الكود ده مكون من 3 Reusable components.  
وكل Component تحتوي على تفاصيل Typescript واسطر من الHtml؛ فكان من الطبيعي يسألني ليه منكتبش فى الComponent بشكل مباشر؛ فجاوبت بالإجابة دي:  
لو لاحظت ان ال3 Forms الي عندنا معمولين بشكل أفضل وكإن Component الlogin مفيهاش غير 2 div بس وده بيدينا 3 حاجات:  
1- الوضوح وتحقيق المطلوب بشكل سريع.  
2- لو حصلت مشكلة هيبقى سهل اننا نحلها.  
3- مش هنتعب في بناء 20 form بالطريقة دي.  
---  
وبالمناسبة ده مثال بسيط جدًا جدًا بالنسبة للReusable component؛ لأنه حرفيًا ملوش حدود وأفتكر اني عملت قبل كدة 4 profile page لنفس البروجيكت بالطريقة دي بتعديلات بسيطة جدًا من خلال الInput() decorator، وبدل ما اتعب نفسي في الTest والتعديل؛ فبقيت اعدل الاربعة من مصدر واحد.